代码说明

一、需求背景

编解码库作为录播转码业务重要的组成部分，编转码，特效，合成，渲染，缩放起着积极的作用，硬编一开始由刘峰打造，随后随着 R400 和 R500 产品的研发，更高码率的画质的需求，硬编的性能问题存在瓶颈，开始考虑加入软编，软合成，特效字幕的处理，原来只用硬编，考虑软硬编码的互通性和复杂性，开始设计软硬转码接口并且统一封装

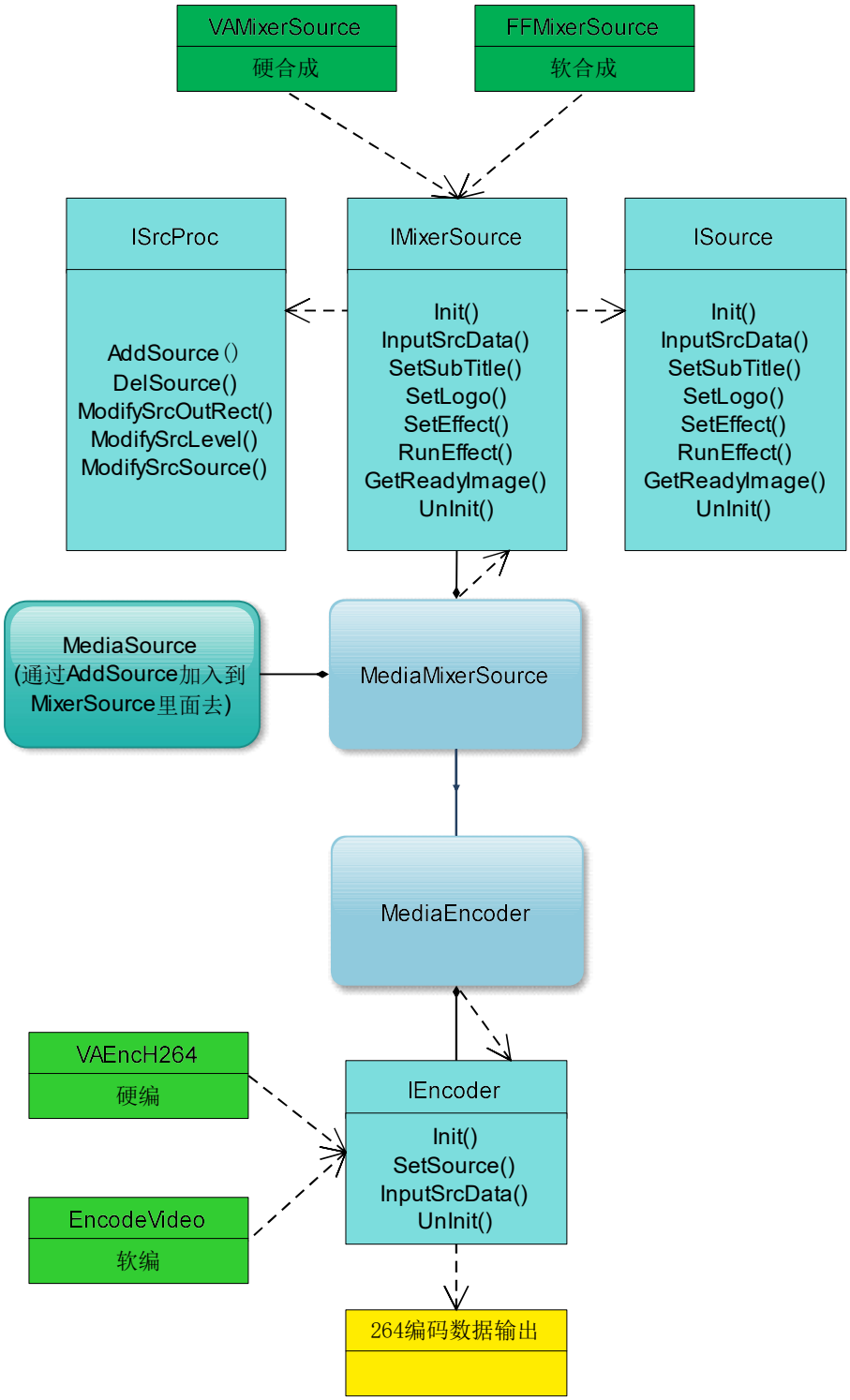
硬处理：使用 intel 内置的 gpu 进行相关处理

软处理：利用 ffmpeg+x264 进行的相关处理

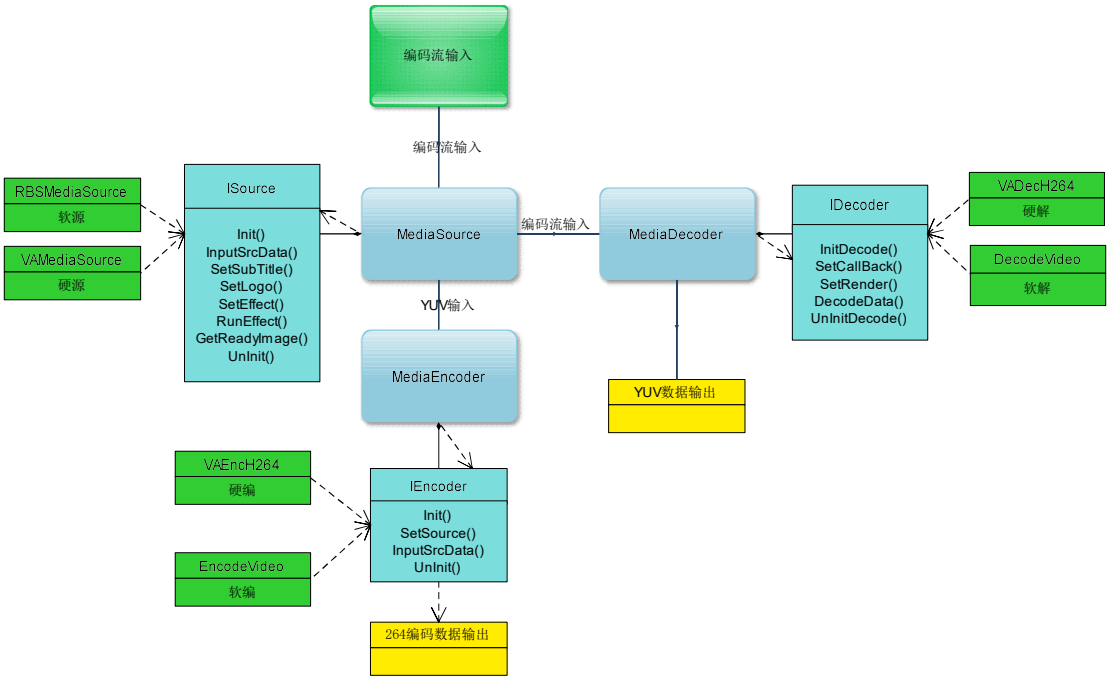
二、设计思路

1、编解码库设计结构

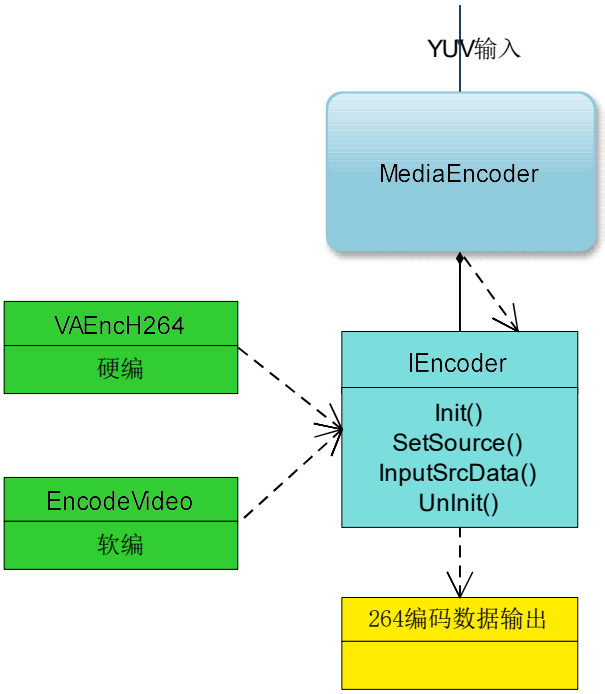
1.合成源设计



2.编码源设计



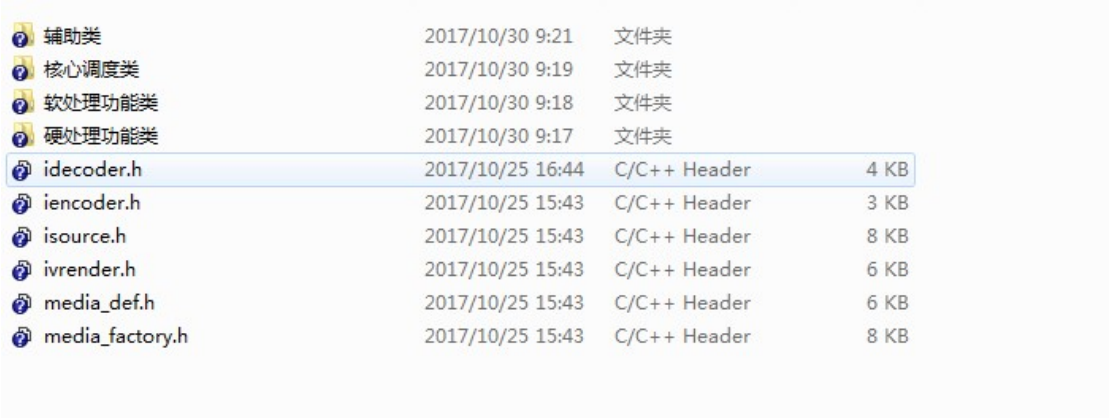
3.YUV 源设计



三、实现说明

3、代码文件说明

目录分层如下图：



编解码涉及到软硬的处理业务和交汇非常复杂，主要文件说明如下：

一、软处理功能类（包含文件如下）：

⑴ decode_video.cpp 软解；

⑵ encode_video.cpp 软编；

⑶ ff_logo.cpp 软叠台标；

⑷ ff_subtitle.cpp 软叠字幕；

⑸ direct_effect.cpp 导播特效；

⑹ ff_media_source.cpp 软处理源

⑺ ff_mixer_render.cpp 软渲染源（先是软合成，在用硬渲染接口）

⑻ ff_mixer_source.cpp 软合成源

二、硬处理功能类（包含如下文件）：

⑴ vamedia_dec_h264.cpp 硬解

⑵ vamedia_enc_h264.cpp 硬编

⑶ vamedia_mixer_source.cpp 硬合成

⑷ vamedia_define.cpp 宏定义 忽略

⑸ vamedia_render_x11.cpp 硬渲染

⑹ vamedia_sei_h264.cpp 忽略

⑺ vamedia_source.cpp 硬处理源

⑻ vamedia_stream_h264.cpp 忽略

三、核心调度类（包含如下文件）（必看）：

media_decoder.cpp 解码口

media_encoder.cpp 编码口

media_factory.cpp 工厂接口（所有创建均从这里开始）

media_render.cpp 渲染口

media_source.cpp 数据源口

四、辅助类（都可以忽略）

media_proc_route.cpp 媒体模式输出（导播源模式和资源模式不同有用）

media_ring_buffer.cpp 环形缓冲 忽略

五、通用接口类（包含以下文件）（必看）
Idecoder.h 解码抽象接口
Iencoder.h 编码抽象接口
Isource.h 数据源抽象接口
Ivrender.h 渲染器抽象接口
media_factor.h 媒体工厂（创建以上所有资源的入口）

四、总结

优点：接口清晰，层次分明，利用”代理模式”隔绝抽象层和实现层，SDK 接口完善，对外友好

缺点：编码器之中用了媒体模式输出 (区分是标准是导播源的软硬处理和资源模式的软硬处理)，导致软硬解码数据回传存在误会，后续会考虑软硬数据的再区分